Temporal Induced Self-Play for Stochastic Bayesian Games

Weizhe Chen[1*], **Zihan Zhou**[2*], **Yi Wu**[2,3] and **Fei Fang**[4]
[1]Shanghai Jiao Tong University
[2]Shanghai Qi Zhi Institute
[3]Tsinghua University
[4]Carnegie Mellon University
chenweizhe@sjtu.edu.cn, {footoredo, jxwuyi}@gmail.com, feif@cs.cmu.edu

Abstract

One practical requirement in solving dynamic games is to ensure that the players play well from any decision point onward. To satisfy this requirement, existing efforts focus on equilibrium refinement, but the scalability and applicability of existing techniques are limited. In this paper, we propose Temporal-Induced Self-Play (TISP), a novel reinforcement learning-based framework to find strategies with decent performances from any decision point onward. TISP uses belief-space representation, backward induction, policy learning, and non-parametric approximation. Building upon TISP, we design a policy-gradient-based algorithm TISP-PG. We prove that TISP-based algorithms can find approximate Perfect Bayesian Equilibrium in zero-sum one-sided stochastic Bayesian games with finite horizon. We test TISP-based algorithms in various games, including finitely repeated security games and a grid-world game. The results show that TISP-PG is more scalable than existing mathematical programming-based methods and significantly outperforms other learning-based methods.

1 Introduction

Many real-world problems involve multiple decision-makers interacting strategically. Over the years, a significant amount of work has focused on building game models for these problems and designing computationally efficient algorithms to solve the games [Serrino *et al.*, 2019; Nguyen *et al.*, 2019]. While Nash equilibrium (NE) is a well-accepted solution concept, the players' behavior prescribed by an NE can be irrational off the equilibrium path: one player can threaten to play a suboptimal action in a future decision point to convince the other players that they would not gain from unilateral deviation. Such "non-credible threats" restrict the practical applicability of these strategies as in the real world, one may make mistakes unexpectedly, and it is hard to enforce such threats. Thus it is important to find strategy profiles such that each player's strategy is close to optimal (in expectation) from any point onward given the other players' strategies.

To find such strategy profiles, researchers have proposed equilibrium refinement concepts such as subgame perfect equilibrium and perfect Bayesian equilibrium (PBE) [Cho and Kreps, 1987] and studied the computational complexity [An *et al.*, 2011; Etessami *et al.*, 2014; Hansen and Lund, 2018]. However, existing methods for computing refined equilibria have limited scalability and often require full access to the game environment, thus can hardly apply to complex games and real-world problems (as detailed in Section 2). On the other hand, deep reinforcement learning (RL) has shown great promise in complex sequential decision-making problems for single-agent and multi-agent settings [Mnih *et al.*, 2015; Silver *et al.*, 2018]. Deep RL leverages a compact representation of the game's state and the players' action space, making it possible to handle large games that are intractable for non-learning-based methods. Despite the promise, to our knowledge, no prior work has applied deep RL to *equilibrium refinements*.

In this paper, we focus on two-player stochastic Bayesian games with finite horizon as they can be used to model various long-term strategic interactions with private information [Albrecht and Ramamoorthy, 2013]. We propose Temporal-Induced Self-Play (TISP), the first RL-based framework to find strategy profiles with decent performances from any decision point onward. There are several crucial challenges in using RL for this task. First, in these games, a player's action at a decision point should be dependent on the entire history of states and joint actions. As the number of histories grows exponentially, a tabular approach that enumerates all the histories is intractable. Although recurrent neural networks (RNNs) can be used to encode the history, RNNs are typically brittle in training and often fail to capture long-term dependency in complex games. Second, using standard RL algorithms with self-play suffers from limited exploration. Hence, it is extremely hard to improve the performance on rarely visited decision points. Our framework TISP tackles these two challenges jointly. We use a belief-based representation to address the first challenge, so that the policy representation remains constant in size regardless of the number of rounds. Besides, we use backward induction to ensure exploration in training. TISP also uses non-parametric approximation in the belief space. Building upon TISP, we design TISP-PG approach that uses policy gradient (PG) for policy learning. TISP can also be combined

[*]Equal Contribution

with other game-solving techniques such as counterfactual regret minimization (CFR) [Zinkevich *et al.*, 2007]. Further, we prove that TISP-based algorithms can find approximate PBE in zero-sum stochastic Bayesian games with one-sided incomplete information and finite horizon. We evaluate TISP-based algorithms in different games. We first test them in finitely repeated security games with unknown attacker types whose PBE can be approximated through mathematical programming (MP) under certain conditions [Nguyen *et al.*, 2019]. Results show that our algorithms can scale to larger games and apply to more general game settings, and the solution quality is much better than other learning-based approaches. We also test the algorithms in a two-step matrix game with a closed-form PBE. Our algorithms can find close-to-equilibrium strategies. Lastly, we test the algorithms in a grid-world game, and the experimental results show that TISP-PG performs significantly better than other methods.

2 Related Work

The study of equilibrium refinements is not new in economics [Kreps and Wilson, 1982]. In addition to the backward induction method for perfect information games, mathematical programming (MP)-based methods [Nguyen *et al.*, 2019; Farina and Gatti, 2017; Miltersen and Sørensen, 2010] have been proposed to compute refined equilibria. However, the MPs used are non-linear and often have an exponential number of variables or constraints, resulting in limited scalability. A few works use iterative methods [Farina *et al.*, 2017; Kroer *et al.*, 2017] but they require exponentiation in game tree traversal and full access to the game structure, which limits their applicability to large complex games.

Stochastic Bayesian games have been extensively studied in mathematics and economics [Forges, 1992; Sorin, 2003; Chandrasekaran *et al.*, 2017; Albrecht and Ramamoorthy, 2013]. [Albrecht *et al.*, 2016] discussed the advantage of using type approximation to approximate the behaviors of agents to what have already been trained, to reduce the complexity in artificial intelligence (AI) researches. We focus on equilibrium refinement in these games and provide an RL-based framework.

Various classical multi-agent RL algorithms [Littman, 1994; Hu *et al.*, 1998] are guaranteed to converge to an NE. Recent variants [Heinrich *et al.*, 2015; Lowe *et al.*, 2017; Iqbal and Sha, 2019] leverage the advances in deep learning [Mnih *et al.*, 2015] and have been empirically shown to find well-performing strategies in large-scale games, such as Go [Silver *et al.*, 2018] and StarCraft [Vinyals *et al.*, 2019]. We present an RL-based approach for equilibrium refinement.

Algorithms for solving large zero-sum imperfect information games like Poker [Moravčík *et al.*, 2017; Brown and Sandholm, 2018] need to explicitly reason about beliefs. Many recent algorithms in multi-agent RL use belief space policy or reason about joint beliefs. These works assume a fixed set of opponent policies that are unchanged [Shen and How, 2019], or consider specific problem domains [Serrino *et al.*, 2019; Woodward *et al.*, 2020]. Foerster *et al.*[2019] uses public belief state to find strategies in a fully cooperative partial information game. We consider stochastic Bayesian games and use belief over opponents' types.

3 Preliminaries

3.1 One-Sided Stochastic Bayesian Game

For expository purposes, we will mainly focus on what we call one-sided stochastic Bayesian games (OSSBG), which extends finite-horizon two-player stochastic games with type information. In particular, player 1 has a private *type* that affects the payoff function. Hence, in a competitive setting, player 1 needs to hide this information, while player 2 needs to infer the type from player 1's actions. Our algorithms can be extended to handle games where both players have types, as we will discuss in Section 6. Formally, an OSSBG is defined by a 8-tuple $\Gamma = \langle \Omega, \mu^0, \Lambda, p^0, \mathcal{A}, P, \{u_i^\lambda\}, L, \gamma \rangle$. Ω is the state space. μ^0 is the initial state distribution. $\Lambda = \{1, \dots, |\Lambda|\}$ is the set of types for player 1. p^0 is the prior over player 1's type. $\mathcal{A} = \prod_i \mathcal{A}_i$ is the joint action space with $\mathcal{A}_i$ the action space for player i. $P : \Omega \times \mathcal{A} \to \Delta_{|\Omega|}$ is the transition function where $\Delta_{|\Omega|}$ represents the $|\Omega|$-dimensional probability simplex. $u_i^\lambda : \Omega \times \mathcal{A} \to \mathbb{R}$ is the payoff function for player i given player 1's type λ. L denotes the length of the horizon or number of rounds. γ is the discount factor.

One play of an OSSBG starts with a type λ sampled from p^0 and an initial state s^0 sampled from μ^0. Then, L *rounds* of the game will rollout. In round l, players take actions $a_1^l \in \mathcal{A}_1$ and $a_2^l \in \mathcal{A}_2$ simultaneously and independently, based on the history $h^l := \{s^0, (a_1^0, a_2^0), \dots, (a_1^{l-1}, a_2^{l-1}), s^l\}$. The players will then get payoff $u_i^\lambda(s^t, a_1^l, a_2^l)$. Note that the payoff $u_i^\lambda(s^l, a_1, a_2)$ at every round l will not be revealed until the end of every play on the game to prevent type information leakage. The states transit w.r.t. $P(s^{l+1}|s^l, a_1^l, a_2^l)$ across rounds.

Let $\mathcal{H}^l$ denote the set of all possible histories in round l and $\mathcal{H} = \bigcup_l \mathcal{H}^l$. Let $\pi_1 : \Lambda \times \mathcal{H} \to \Delta_{\mathcal{A}_1}, \pi_2 : \mathcal{H} \to \Delta_{\mathcal{A}_2}$ be the players' behavioral strategies. Given the type λ, the history h^l and the strategy profile $\pi = (\pi_1, \pi_2)$, player i's discounted accumulative expected utility from round l onward is

$$\mathrm{V}_i^\lambda(\pi, h^l) = \sum_{a_1, a_2} \Big(\pi_1(a_1|\lambda, h^l)\pi_2(a_2|h^l) \Big(u_i^\lambda(s^l, a_1, a_2) + \gamma \sum_{s'} P(s'|s^l, a_1, a_2)\, \mathrm{V}_i^\lambda(\pi, h^l \cup \{(a_1, a_2), s'\}) \Big) \Big). \quad (1)$$

Similarly, we can define the Q function by $Q_i^\lambda(\pi, h^l, a) = u_i^\lambda(s^l, a_1, a_2) + \gamma \mathbb{E}_{s'} \left[V_i^\lambda(\pi, h^l \cup \{(a_1, a_2), s'\}) \right]$.

An OSSBG can be converted into an equivalent extensive-form game (EFG) with imperfect information where each node in the game tree corresponds to a (type, history) pair (see Appendix F). However, this EFG is exponentially large, and existing methods for equilibrium refinement in EFGs [Kroer *et al.*, 2017] are not suitable due to their limited scalability.

3.2 Equilibrium Concepts in OSSBG

Let Π_i denote the space of all valid strategies for Player i.

Definition 3.1 (ϵ-NE) *A strategy profile $\pi = (\pi_1, \pi_2)$ is an ϵ-NE if for $i = 1, 2$, and all visitable history h^l corresponding to the final policy,*

$$\max_{\pi_i' \in \Pi_i} \mathrm{V}_i(\pi_i|_{h^l \to \pi_i'}, \pi_{-i}, h^l) - \mathrm{V}_i(\pi, h^l) \leq \epsilon \qquad (2)$$

where $\pi_i|_{h^l \to \pi_i'}$ means playing π_i until h^l is reached, then playing π_i' onwards.

Definition 3.2 (ϵ-PBE) *A strategy profile $\pi = (\pi_1, \pi_2)$ is an ϵ-PBE if for $i = 1, 2$ and all histories h^l,*

$$\max_{\pi_i' \in \Pi_i} \mathrm{V}_i(\pi_i|_{h^l \to \pi_i'}, \pi_{-i}, h^l) - \mathrm{V}_i(\pi, h^l) \leq \epsilon \qquad (3)$$

where $\pi_i|_{h^l \to \pi_i'}$ means playing π_i until h^l is reached, then playing π_i' onwards.

It is straightforward that an ϵ-PBE is also an ϵ-NE.

4 Temporal-Induced Self-Play

Our TISP framework (Alg. 1) considers each player as an RL agent and trains them with self-play. Each agent maintains a policy and an estimated value function, which will be updated during training. TISP has four ingredients. It uses belief-based representation (Sec. 4.1), backward induction (Sec. 4.2), policy learning (Sec. 4.3) and belief-space approximation (Sec. 4.4). We discuss test-time strategy and show that TISP converges to ϵ-PBEs under certain conditions in Sec. 4.5.

4.1 Belief-Based Representation

Instead of representing a policy as a function of the history, we consider player 2's belief $b \in \Delta_{|\Lambda|}$ of player 1's type and represent π_i as a function of the belief b and the current state s^l in round l, i.e., $\pi_{1,l}(\cdot|b, s^l, \lambda)$ and $\pi_{2,l}(\cdot|b, s^l)$. The belief b represents the posterior probability distribution of λ and can be obtained using Bayes rule given player 1's strategy:

$$b_\lambda^{l+1} = \frac{\pi_{1,l}\left(a_1^l|s^l, b_\lambda^l, \lambda\right) b_\lambda^l}{\sum_{\lambda' \in \Lambda} \pi_{1,l}\left(a_1^l|s^l, b_{\lambda'}^l, \lambda'\right) b_{\lambda'}^l} \qquad (4)$$

where b_λ^l is the probability of player 1 being type λ given all its actions up to round $l-1$. This belief-based representation avoids the enumeration of the exponentially many histories. Although it requires training a policy that outputs an action for any input belief in the continuous space, it is possible to use approximation as we show in Sec. 4.4. We can also define the belief-based value function for agent i in round l by

$$V_{i,l}^\lambda(\pi, b^l, s^l) = \mathbb{E}_{a_1, a_2, s^{l+1}}\big[u_i^\lambda(s^l, a_1, a_2) + \gamma V_{i,l+1}^\lambda(\pi, b^{l+1}, s^{l+1})\big]. \qquad (5)$$

The Q-function $Q_{i,l}^\lambda(\pi, b^l, s^l, a^l)$ can be defined similarly. We assume the policies and value functions are parameterized by θ and ϕ respectively with neural networks.

4.2 Backward Induction

Standard RL approaches with self-play train policies in a top-down manner: it executes the learning policies from round 0 to $L-1$ and only learns from the experiences at visited decision points. To find strategy profiles with decent performances from any decision point onward, we use backward induction and train the policies and calculate the value functions in the reverse order of rounds: we start by training $\pi_{i,L-1}$ for all agents and then calculate the corresponding value functions $V_{i,L-1}^\lambda$, and then train $\pi_{i,L-2}$ and so on.

The benefit of using backward induction is two-fold. In the standard forward-fashioned approach, one needs to roll out the entire trajectory to estimate the accumulative reward for policy and value learning. In contrast, with backward induction, when training $\pi_{i,l}$, we have already obtained $V_{i,l+1}^\lambda$. Thus, we just need to roll out the policy for 1 round and directly estimate the expected accumulated value using $V_{i,l+1}^\lambda$ and Eq. (5). Hence, we effectively reduce the original L-round game into L 1-round games, which makes the learning much easier. Another important benefit is that we can uniformly sample all possible combinations of state, belief and type at each round to ensure effective exploration. More specifically, in round l, we can sample a belief b and then construct a new game by resetting the environment with a uniformly randomly sampled state and a type sampled from b. Implementation-wise, we assume access to an auxiliary function from the environment, called *sub_reset*(l, b) that produces a new game as described. This function takes two parameters, a round l and a belief b, as input and produces a new game by drawing a random state from the entire state space Ω with equal probability and a random type according to the belief distribution b. This function is an algorithmic requirement for the environment, which is typically feasible in practice. For example, most RL environments provides a *reset* function that generates a random starting state, so a simple code-level enhancement on this *reset* function can make existing testbeds compatible with our algorithm. We remark that even with such a minimal environment enhancement requirement, our framework does *NOT* utilize the transition information. Hence, our method remains nearly model-free comparing to other methods that assume full access to the underlying environment transitions — this is the assumption of most CFR-based algorithms. Furthermore, using customized reset function is not rare in the RL literature. For example, most automatic curriculum learning algorithms assumes a flexible reset function that can reset the state of an environment to a desired configuration.

4.3 Policy Learning

Each time a game is produced by *sub_reset*(l, b), we perform a 1-round learning with self-play to find the policies $\pi_{i,l}$. TISP allows different policy learning methods. Here we consider two popular choices, policy gradient and regret matching.

Policy Gradient

PG method directly takes a gradient step over the expected utility. For notational conciseness, we omit the super/subscripts of i, l, λ in the following equations and use s,

b and s', b' to denote the state and belief at current round and next round.

Theorem 4.1 *In the belief-based representation, the policy gradient derives as follows:*

$$\nabla_\theta V^\lambda(\pi, b, s) = \sum_{a\in\mathcal{A}} \nabla_\theta \left(\pi_\theta(a|b,s) Q^\lambda(\pi, b, s, a)\right) \quad (6)$$
$$= \mathbb{E}_{a,s'}\Big[Q^\lambda(\pi, b, s, a)\nabla_\theta \ln \pi_\theta(a|b,s) + \gamma \nabla_\theta b' \nabla_{b'} V^\lambda(\pi, b', s')\Big].$$

Comparing to the standard policy gradient theorem, we have an additional term in Eq.(6) (the second term). Intuitively, when the belief space is introduced, the next belief b' is a function of the current belief b and the policy π_θ in the current round (Eq.(4)). Thus, the change in the current policy may influence future beliefs, resulting in the second term. The full derivation can be found in Appendix E. We also show in the experiment section that when the second term is ignored, the learned policies can be substantially worse. We refer to this PG variant of TISP framework as TISP-PG.

Regret Matching

Regret matching is another popular choice for imperfect information games. We take inspirations from Deep CFR [Brown *et al.*, 2019] and propose another variant of TISP, referred to as TISP-CFR. Specifically, for each training iteration t, let $R^t(s,a)$ denote the regret of action a at state s, $\pi^t(a|s,b)$ denote the current policy, $Q^t(\pi^t, s, b, a)$ denote the Q-function and $V^t_\phi(\pi^t, s, b)$ denote the value function corresponding to π^t. Then we have

$$\pi^{t+1}(a|s,b) \quad = \quad \frac{(R^{t+1}(s,b,a))^+}{\sum_{a'}(R^{t+1}(s,b,a'))^+},$$

where $R^{t+1}(s,b,a) = \sum_{\tau=1}^t Q^\tau(\pi^\tau, s, b, a) - V^\tau_\phi(\pi^\tau, s, b)$ and $(\cdot)^+ := \max(\cdot, 0)$. Since the policy can be directly computed from the value function, we only learn V^t_ϕ here. Besides, most regret-based algorithms require known transition functions, which enables them to reset to any infomation-set node in the training. We use the outcome sampling method [Lanctot *et al.*, 2009], which samples a batch of transitions to update the regret. This is similar to the value learning procedure in standard RL algorithms. This ensures our algorithm to be model-free. Although TISP-CFR does not require any gradient computation, it is in practice much slower than TISP-PG since it has to learn an entire value network in every single iteration.

4.4 Belief-Space Policy Approximation

A small change to the belief can drastically change the policy. When using a function approximator for the policy, this requires the network to be sensitive to the belief input. We empirically observe that a single neural network often fails to capture the desired belief-conditioned policy. Therefore, we use an ensemble-based approach to tackle this issue.

At each round, we sample K belief points $\{b_1, b_2, ..., b_K\}$ from the belief space $\Delta^{|\Lambda|}$. For each belief b_k, we use self-play to learn an accurate independent strategy $\pi_{\theta_k}(a|s;b_k)$ and value $V_{\phi_k}(\pi, s; b_k)$ over the state space but specifically conditioning on this particular belief input b_k. When querying the policy and value for an arbitrary belief b different from the sampled ones, we use a distance-based non-parametric method to approximate the target policy and value. Specifically, for any two belief b_1 and b_2, we define a distance metric $w(b_1, b_2) = \frac{1}{\max(\epsilon, \|b-b'\|^2)}$ and then for the query belief b, we calculate its policy $\pi(a|b,s)$ and value $V(\pi, b, s)$ by

$$\pi(a|b,s) \quad = \quad \frac{\sum_{k=1}^K \pi_{\theta_k}(a|s;b_k) w(b, b_k)}{\sum_{k=1}^K w(b, b_k)} \quad (7)$$

$$V(\pi, b, s) \quad = \quad \frac{\sum_{k=1}^K V_{\phi_k}(\pi, s; b_k) w(b, b_k)}{\sum_{k=1}^K w(b, b_k)} \quad (8)$$

We introduce a density parameter d to ensure the sampled points are dense and has good coverage over the belief space. d is defined as the farthest distance between any point in $\Delta^{|\Lambda|}$ and its nearest sampled points, or formally $d = \sup\{\min\{\|b, b_i\| \mid i = 1, \ldots, K\} \mid b \in \Delta^{|\Lambda|}\}$. A smaller d indicates a denser sampled points set.

Note that the policy and value networks at different belief points at each round are completely independent and can be therefore trained in perfect parallel. So the overall training wall-clock time remains unchanged with policy ensembles. Additional discussions on belief sampling are in Appx. A.

4.5 Test-time Strategy

Note that our algorithm requires computing the precise belief using Eq. 4, which requires the known policy for player 1, which might not be feasible at test time when competing against an unknown opponent. Therefore, at test time, we update the belief according to the *training policy* of player 1,

Algorithm 1 Temporal-Induced Self-Play

for $l = L-1, \ldots, 0$ **do**
  for $k = 1, 2, \ldots K$ **do** ▷ run in parallel
    for $t = 1, \ldots, T$ **do**
      Initialize replay buffer $D = \{\}$ and π^0
      for $j = 1, \ldots,$ batch size **do** ▷ parallel
        $s \leftarrow sub\_reset(l, b_k)$;
        $a \leftarrow \pi^{t-1}_{\theta_{l,k}}(s; b_k)$;
        get next state s' and utility u from env;
        $D \leftarrow D + (s, a, s', u)$;
      Update $V^t_{\phi_{l,k}}$ and $\pi^t_{\theta_{l,k}}$ using D;
    $V_{\phi_{l,k}} \leftarrow V^n_{\phi_{l,k}}$, $\pi_{\theta_{l,k}} \leftarrow \pi^n_{\theta_{l,k}}$
return $\{\pi_{\theta_{l,k}}, V_{\phi_{l,k}}\}_{0\le l<L, 1\le k\le K}$;

Algorithm 2 Compute Test-Time Strategy

function GETSTRATEGY($h^l, \pi_{\theta_1}, \ldots, \pi_{\theta_L}$)
  $b^0 \leftarrow p^0$
  for $j \leftarrow 0, \ldots, l-1$ **do**
    update b^{j+1} using b^j, s^j, a^j and π_{θ_j} with Eq.(4)
  return $\pi(a|b^l, s^l)$ with Eq.(7)

regardless of its actual opponent policy. That is, even though the actions produced by the actual opponent can be completely different from the oracle strategies we learned from training, we still use the oracle policy from training to compute the belief. Note that it is possible that we obtain an infeasible belief, i.e., the opponent chooses an action with zero probability in the oracle strategy. In this case, we simply use a uniform belief instead. The procedure is summarized in Alg. 2. We also theoretically prove in Thm. 4.2 that in the zero-sum case, the strategy provided in Alg. 2 provides a bounded approximate PBE and further converges to a PBE with infinite policy learning iterations and sampled beliefs. The poof can be found in Appendix F.

Theorem 4.2 *When the game is zero-sum and Ω is finite, the strategies produced by Alg. 2 is ϵ-PBE, where $\epsilon = L(dU + c \cdot T^{-1/2})$, d is the distance parameter in belief sampling, $U = L \cdot \big(\max_{i,s,a_1,a_2} u_i(s,a_1,a_2) - \min_{i,s,a_1,a_2} u_i(s,a_1,a_2)\big)$, n is the number of iterations in policy learning and c is a positive constant associated with the particular algorithm (TISP-PG or TISP-CFR, details in Appendix F). When $T \to \infty$ and $d \to 0$, the strategy becomes a PBE.*

We remark that TISP is nearly-model-free and does not utilize the transition probabilities, which ensures its adaptability.

5 Experiment

We test our algorithms in three sets of games. While very few previous works in RL focus on equilibrium refinement, we compare our algorithm with self-play PG with RNN-based policy (referred to as RNN) and provide ablation studies for the TISP-PG algorithm: *BPG* uses only belief-based policy without backward induction or belief-space approximation; *BI* adopt backward induction and belief-based policy but does not use belief-space approximation. Full experiment details can be found in Appx. D.

5.1 Finitely Repeated Security Game

Game Setting

We consider a finitely repeated *simultaneous-move* security game, as discussed in [Nguyen *et al.*, 2019]. Specifically, this is an extension of a one-round security game by repeating it for L rounds. Each round's utility function can be seen as a special form of matrix game and remains the same across rounds. In each round, the attacker can choose to attack one position from all A positions, and the defender can choose to defend one. The attacker succeeds if the target is not defended. The attacker will get a reward if it successfully attacks the target and a penalty if it fails. Correspondingly, the defender gets a penalty if it fails to defend a place and a reward otherwise. In the zero-sum setting, the payoff of the defender is the negative of the attacker's. We also adopt a general-sum setting described in [Nguyen *et al.*, 2019] where the defender's payoff is only related to the action it chooses, regardless of the attacker's type.

Evaluation

We evaluate our solution by calculating the minimum ϵ so that our solution is an ϵ-PBE. We show the average result of 5 different game instances.

	TISP-PG	RNN	BPG	BI
Mean ϵ	**0.881**	15.18	101.2	27.51
Worst ϵ	**1.220**	31.81	111.8	42.54

(a) Zero-sum result for model-free methods, with $|\Lambda| = 2$.

	TISP-PG	RNN	BPG	BI
Mean ϵ	**0.892**	34.62	89.21	83.00
Worst ϵ	**1.120**	57.14	182.1	111.9

(b) General-sum result for model-free methods, with $|\Lambda| = 2$.

	Zero-sum		General-sum	
	TISP-PG	TISP-CFR	TISP-PG	TISP-CFR
Mean ϵ	**0.446**	0.474	**0.608**	0.625
Worst ϵ	**1.041**	1.186	**1.855**	1.985

(c) Result for known model variants, with $|\Lambda| = 2$.

	TISP-PG	RNN	BPG	BI
Mean ϵ	**1.888**	18.20	79.74	40.75
Worst ϵ	**3.008**	28.15	97.67	49.74

(d) Zero-sum result, with $|\Lambda| = 3$.

Table 1: The result for finitely repeated security game. The less the number, the better the solution is. These results are evaluated with $L = 10$, $|A| = 2$, and uniform prior distribution.

L	2	4	6	8	10
MP	$\approx 10^{-8}$	$\approx 10^{-6}$	$\approx 10^{-5}$	N/A	N/A
TISP-PG	0.053	0.112	0.211	0.329	0.473
TISP-CFR	0.008	0.065	0.190	0.331	0.499

(a) $|A| = 2$

L	2	4	6	8	10
MP	$\approx 10^{-6}$	$\approx 10^{-6}$	$\approx 10^{-3}$	N/A	N/A
TISP-PG	0.120	0.232	0.408	0.599	0.842
TISP-CFR	0.002	0.049	0.285	0.525	0.847

(b) $|A| = 5$

Table 2: Comparing mathematical-programming and our methods, i.e., TISP-PG and TISP-CFR, with known model. These results are averaged over 21 starting prior distributions of the attacker ($[0.00, 1.00], [0.05, 0.95], \ldots, [1.00, 0.00]$).

Results

We first experiment with the zero-sum setting where we have proved our model can converge to an ϵ-PBE. The comparison are shown in Table 1a,1c. We use two known model variants, TISP-PG and TISP-CFR, and a model-free version of TISP-PG in this comparison. TISP-PG achieves the best results, while TISP-CFR also has comparable performances. We note that simply using an RNN or using belief-space policy performs only slightly better than a random policy.

Then we conduct experiments in general-sum games with results shown in Table 1b,1c. We empirically observe that the derived solution has comparable quality with the zero-sum setting. We also compare our methods with the Mathematical-Programming-based method (MP) in [Nguyen *et al.*, 2019], which requires full access to the game transition. The results are shown in Table 2. Although when L is small, the MP solution achieves superior accuracy, it quickly runs out of memory (marked as "N/A") since its time and memory requirement grows at an exponential rate w.r.t. L. Again, our TISP variants perform the best among all learning-based methods. We remark that despite of the performance gap be-

tween our approach and the MP methods, the error on those games unsolvable for MP methods is merely 0.1% comparing to the total utility.

In our experiments, we do not observe orders of magnitudes difference in running time between our method and baselines: TISP-PG and TISP-CFR in the Tagging game uses 20 hours with 10M samples in total even with particle-based approximation (200k samples per belief point) while RNN and BPG in the Tagging game utilizes roughly 7 hours with 2M total samples for convergence.

Regarding the scalability on the number of types, as the number of types increases, the intrinsic learning difficulty increases. This is a challenge faced by all the methods. The primary contribution of this paper is a new learning-based framework for PBNE. While we primarily focus on the case of 2 types, our approach generalizes to more types naturally with an additional experiment conducted for 3 types in Table. 1d. Advanced sampling techniques can potentially be utilized for more types, which we leave for future work.

5.2 Exposing Game

Game Setting

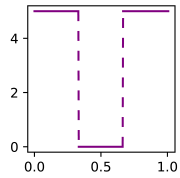
We also present a two-step matrix game, which we call *Exposing*. In this game, player 2 aims to guess the correct type of player 1 in the second round. There are two actions available for player 1 and three actions available for player 2 in each round. Specifically, the three actions for player 2 means guessing player 1 is *type 1*, *type 2* or not guessing at all. The reward for a correct and wrong guess is 10 and -20 respectively. The reward for not guessing is 0. Player 1 receives a positive 5 reward when the player 2 chooses to guess in the second round, regardless of which type player 2 guesses. In this game, player 1 has the incentive to expose its type to encourage player 2 to guess in the second round. We further add a reward of 1 for player 1 choosing action 1 in the first round regardless of its type to give player 1 an incentive to not exposing its type. With this reward, a short-sighted player 1 may pursue the 1 reward and forgo the 5 reward in the second round. The payoff matrices for this game are in Appx. D.

The equilibrium strategy for player 2 in the second round w.r.t. different types of player 1 is:

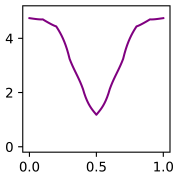
$$\text{strategy} := \begin{cases} \text{guessing type 1} & \text{if } \Pr[1|h] > \frac{1}{3} \\ \text{guessing type 2} & \text{if } \Pr[2|h] > \frac{1}{3} \\ \text{not guessing} & \text{else} \end{cases}$$

In this game, the equilibrium values in the second round for both types of player 1 are highly discontinuous with regard to player 2's belief, as shown in Fig. 1a, which makes the belief-space gradient term in Eq. 6 ineffective. However, the approximation introduced in Sec. 4.4 can serve as a soft representation of the true value function, as shown in Fig. 1b. This soft representation provides an approximated gradient in the belief space, which allows for belief-space exploration. We will show later that this belief-space exploration cannot be achieved otherwise.

[1]The optimal solution refers to the Pareto-optimal PBE in this game. Note that there are two symmetric optimal solutions. We choose to only show one here for easy comparison and simplicity.



(a) Ground truth



(b) Approximation

Figure 1: Ground truth and approximated value of player 1 in the second round of Exposing game. The $x-$axis corresponds to $\Pr[1|h]$. The $y-$axis corresponds to the equilibrium value.

	P1's type	Action 1	Action 2	Reward
TISP-PG	type 1	0.985	0.015	5.985
	type 2	0.258	0.742	5.258
TISP-CFR	type 1	1.000	0.000	1.000
	type 2	1.000	0.000	1.000
TISP-PG$^-$	type 1	0.969	0.031	0.969
	type 2	0.969	0.031	0.969
Optimal	type 1	1.000	0.000	6.000
	type 2	0.333	0.667	5.333

Table 3: Detailed first round policy in Exposing game. TISP-PG is the only algorithm that separates the two player-1 types' strategies and yields a result very close to the optimal solution.[1]

Results

We compare the training result between TISP-PG and TISP-CFR to exhibit the effectiveness of our non-parametric approximation. We further add an ablation study that removes the belief-space gradient term in TISP-PG, which we call TISP-PG$^-$. The results are shown in Table 3. We can see that TISP-PG is the only algorithm that successfully escapes from the basin area in Fig. 1 as it is the only algorithm that is capable of doing belief-space exploration. We also show the obtained policies Table 3. The training curves can be found in Appx. D. Note that the policies trained from TISP-CFR and TISP-PG$^-$ are also close to a PBE where player 1 takes action 1 regardless of its type in the first round, and player 2 chooses not to guess in the second round, although it is not Pareto-optimal.

5.3 Tagging Game

Game Setting

We test our method in a gridworld game *Tagging*, as illustrated in Fig. 2. The game is inspired by [Shen and How, 2019]. Specifically, the game is on an 8×8 square, and player 1 has two types, i.e., *ally* and *enemy*, and each type corresponds to a unique target place. Each player will receive distance-based reward to encourage it to move towards its target place. There is a river in the top half part of the grids which Player 2 cannot enter. Player 1 starts from the bottom middle of the map and player 2 starts from a random position under the river. Both players can choose to move in one of the four directions, [up(U), down(D), left(L), right(R)], by one cell. Player 2 has an additional action, *tag*, to tag player 1 as the *enemy* type. The tag action is only available when player 1 has not entered the river and the euclidean distance between the two players is less than 2.5. The attackers get

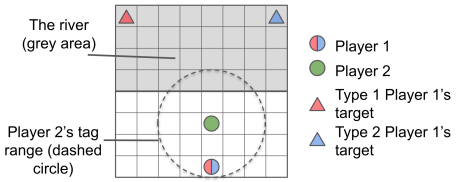


Figure 2: An illustration of the Tagging game.

	P1's Type	U	D	R	L
TISP-PG	Ally	0.839	0.001	0.001	0.159
	Enemy	0.932	0.001	0.001	0.066
RNN	Ally	0.248	0.237	0.238	0.274
	Enemy	0.599	0.067	0.077	0.255
BPG	Ally	0.000	0.000	0.000	1.000
	Enemy	1.000	0.000	0.000	0.000
TISP-CFR	Ally	0.000	0.000	0.000	1.000
	Enemy	1.000	0.000	0.000	0.000

Table 4: The policy at one of the starting states in Tagging game, where player 2 is two cells above the fixed starting point of player 1.

higher rewards for getting closer to their type-specified target, and the defenders get higher rewards for getting closer to the attacker. Moreover, if the defender chooses to tag, it will get a reward of 10 if the attacker is of the *enemy* type and a reward of -20 if the attacker is of the *ally* type, while the attacker will get a reward of -10 for being tagged no matter what its type is. More detail is provided in Appx. D.

Based on the game's rule, an enemy-typed player 1 is likely to go to its own target immediately to get a higher reward. However, such a strategy reveals its type straight away, which can be punished by being tagged. A more clever strategy of an enemy-typed player 1 is to mimic the behavior of the ally-typed player 1 in most situations and never let player-2's belief of enemy type be larger than $\frac{2}{3}$, so that player 2 has no incentive to tag it. The ally-typed player 1 may simply go up in most states in order to get closer to its target for a reward while player 2 should learn to tag player 1 if its belief of enemy type is high enough and try to move closer to player 1 in other cases.

Evaluation

Although it is computationally intractable to calculate the exact exploitability in this gridworld game, we examine the results following [Gray *et al.*, 2021] by evaluating the performances in induced games. We choose 256 induced games among all the induced games after the second round and check their exploitability. We get the best response in each induced game in two ways: in the first round of each game, we enumerate all the possible actions and manually choose the best action. We also train a new BPG player-2 from scratch. This new BPG-based player is trained in a single-agent-like environment and does not need to consider the change in the player-1 policy. We check how much reward agents can get after the learning process finishes. A high reward from player 2 would indicate that player 1 fail to hide its type. We also provide the detailed strategies from different baselines for the first round of the game for additional insight.

	TISP-PG	RNN	BPG	TISP-CFR
P2 reward	-1.90	-1.67	-0.98	-1.82
P1 reward (ally)	-2.55	-2.87	-3.26	-3.17
P1 reward (enemy)	-2.41	-2.71	-9.29	-4.49

Table 5: The average exploitability result of 256 induced games in the Tagging game. The lower player 2's reward, the better the algorithm.

Results

We show the derived policy in the very first step in Table 4. The policy learned by our method successfully keeps the belief to be less than $\frac{1}{3}$, and keeps a large probability of going to the target of each type. The RNN policy shows no preference between different actions, resulting in not being tagged but also not getting a larger reward for getting closer to the target. The BPG policy simply goes straight towards the target and is therefore punished for being tagged. The exploitability results are shown in Table 5. From the training reward achieved by the new exploiter player 2, TISP-PG performs the best among all baselines and TISP-CFR also produces a robust player-1 policy. We remark that relative performances of different methods in the gridworld game are consistent with what we have previously observed in the finitely repeated security games, which further validates the effectiveness of our approach.

6 Discussion and Conclusion

We proposed TISP, an RL-based framework to find strategies with a decent performance from any decision point onward. We provided theoretical justification and empirically demonstrated its effectiveness. Our algorithms can be easily extended to a two-sided stochastic Bayesian game. The TISP framework still applies, and the only major modification needed is to add a for-loop to sample belief points for player 2. This extension will cost more computing resources, but the networks can be trained fully in parallel. The full version of the extended algorithm is in Appendix C.

Acknowledgements

We would like to thank Ryan Shi for some help in writing the early workshop version of this paper. Co-author Fang is supported in part by NSF grant IIS- 1850477, a research grant from Lockheed Martin, and by the U.S. Army Combat Capabilities Development Command Army Research Laboratory Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies.

References

[Albrecht and Ramamoorthy, 2013] Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *AAMAS*, 2013.

[Albrecht *et al.*, 2016] Stefano V Albrecht, Jacob W Crandall, and Subramanian Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 2016.

[An *et al.*, 2011] Bo An, Milind Tambe, Fernando Ordonez, Eric Shieh, and Christopher Kiekintveld. Refinement of strong stackelberg equilibria in security games. In *AAAI*, 2011.

[Brown and Sandholm, 2018] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 2018.

[Brown *et al.*, 2019] N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. *ICML*, 2019.

[Chandrasekaran *et al.*, 2017] Muthukumaran Chandrasekaran, Yingke Chen, and Prashant Doshi. On markov games played by bayesian and boundedly-rational players. In *AAAI*, 2017.

[Cho and Kreps, 1987] In-Koo Cho and David M Kreps. Signaling games and stable equilibria. *QJE*, 1987.

[Etessami *et al.*, 2014] Kousha Etessami, Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. The complexity of approximating a trembling hand perfect equilibrium of a multi-player game in strategic form. In *SAGT*, 2014.

[Farina and Gatti, 2017] Gabriele Farina and Nicola Gatti. Extensive-form perfect equilibrium computation in two-player games. In *AAAI*, 2017.

[Farina *et al.*, 2017] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Regret minimization in behaviorally-constrained zero-sum games. In *ICML*, 2017.

[Foerster *et al.*, 2019] Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In *ICML*, 2019.

[Forges, 1992] Francoise Forges. Repeated games of incomplete information: non-zero-sum. *Handbook of game theory with economic applications*, 1992.

[Gray *et al.*, 2021] Jonathan Gray, Adam Lerer, Anton Bakhtin, and Noam Brown. Human-level performance in no-press diplomacy via equilibrium search. *ICLR*, 2021.

[Hansen and Lund, 2018] Kristoffer Arnsfelt Hansen and Troels Bjerre Lund. Computational complexity of proper equilibrium. In *EC*, 2018.

[Heinrich *et al.*, 2015] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *ICML*, 2015.

[Hu *et al.*, 1998] Junling Hu, Michael P Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. In *ICML*, 1998.

[Iqbal and Sha, 2019] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*, 2019.

[Kreps and Wilson, 1982] David M Kreps and Robert Wilson. Sequential equilibria. *Econometrica*, 1982.

[Kroer *et al.*, 2017] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. Smoothing method for approximate extensive-form perfect equilibrium. *arXiv*, 2017.

[Lanctot *et al.*, 2009] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In *NIPS*, 2009.

[Littman, 1994] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceeding*. 1994.

[Lowe *et al.*, 2017] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NIPS*, 2017.

[Miltersen and Sørensen, 2010] Peter Bro Miltersen and Troels Bjerre Sørensen. Computing a quasi-perfect equilibrium of a two-player game. *Economic Theory*, 2010.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[Moravčík *et al.*, 2017] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, et al. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 2017.

[Nguyen *et al.*, 2019] Thanh H. Nguyen, Yongzhao Wang, Arunesh Sinha, and Michael P. Wellman. Deception in finitely repeated security games. *AAAI*, 2019.

[Serrino *et al.*, 2019] Jack Serrino, Max Kleiman-Weiner, David C Parkes, and Josh Tenenbaum. Finding friend and foe in multi-agent games. In *Neurips*, 2019.

[Shen and How, 2019] Macheng Shen and Jonathan P How. Robust opponent modeling via adversarial ensemble reinforcement learning in asymmetric imperfect-information games. *arXiv*, 2019.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 2018.

[Sorin, 2003] Sylvain Sorin. Stochastic games with incomplete information. In *Stochastic Games and applications*. 2003.

[Vinyals *et al.*, 2019] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 2019.

[Woodward *et al.*, 2020] Mark Woodward, Chelsea Finn, and Karol Hausman. Learning to interactively learn and assist. *AAAI*, 2020.

[Zinkevich *et al.*, 2007] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. *NIPS*, 2007.